is to make the transmitted codeword, S, Toeplitz, thereby guaranteeing full spatio-temporal diversity. A typical codeword is shown above for the case of $M_T = 2$, $M_R = 1$, $L = 2$, $N = 4$. This code exploits full, fourth-order diversity in the system. Fig. 1 compares the performance of the SDD and GDD codes. The channel taps are assumed to be uncorrelated across space and time implying a potential diversity order of four. Each transmitted frame is 130 symbols long. Clearly, the SDD fails to exploit full diversity gain, whereas the GDD has higher diversity order. The comparison may seem unfair because the GDD code with higher code delay has more states as compared to the SDD code. However, our goal in comparing the two schemes is to show that the extra states do not merely buy coding gain but help in exploiting full diversity as is clear from the steeper FER slope.

Conclusions: We have presented a framework for analysing space-time codes for a delay spread channel. We have proposed a generalised delay diversity code with code delay matched to the delay spread which exploits full spatio-temporal diversity.

20 June 2001
Electronics Letters Online No: 20010860
DOI: 10.1049/el:20010860

D. Gore, S. Sandhu and A. Paulraj (*Packard 225, Information Systems Laboratory, Stanford University, Stanford, CA 94305-9510, USA*)

E-mail: dagore@stanford.edu

References

1 AGRAWAL, D., TAROKH, V., NAGUIB, A., and SESHADRI, N.: 'Space-time coded OFDM for high data-rate wireless communication over wideband channels'. Proc. 48th VTC, 1998, Vol. 3, pp. 2232–2236

2 GONG, Y., and LETAIEF, K.B.: 'Performance evaluation and analysis of space-time coding in unequalized multipath fading links', *IEEE Trans. Commun.*, 2000, **48**, (11), pp. 1778–1782

3 LINDSKOG, E., and PAULRAJ, A.: 'A transmit diversity scheme for channels with intersymbol interference'. Proc. ICC, 2000, Vol. 1, pp. 307–311

4 BOLCSKEI, H., and PAULRAJ, A.: 'Space-frequency codes for broadband fading channels'. ISIT, 2001

5 SESHADRI, N., and WINTERS, J.H.: 'Two signaling schemes for improving the error performance of frequency-division-duplex (FDD) transmission systems using transmitter antenna diversity'. Proc. 43rd VTC, 1993, pp. 508–511

Efficient implementation technique of LDPC decoder

W.K. Leung, W.L. Lee, Angus Wu and Li Ping

An efficient implementation technique of the low density parity check code decoder is proposed. Using the technique, the decoder can be implemented with additions only and there is considerably lower complexity compared with the standard sum-product decoder.

Introduction: The low density parity check (LDPC) code [1] can be decoded by the iterative sum-product algorithm [2]. The latter involves manipulation of the difference between likelihood values which is sensitive to the quantisation effect. It has been shown that the parity-likelihood-ratio (PLR) technique [3] is less sensitive to quantisation.

In this Letter we will show that the PLR technique described in [3] can be efficiently implemented with additions only. The performance can be maintained by using a very simple correction term (either +1 or –1), which is similar to the correction method introduced in [4].

PLR method for decoding LDPC codes: The iterative decoding procedure of the LDPC code described in [3] is summarised below. Let $\mathbf{c} = \{c_n\}$ be the transmitted codeword over $\{+1, -1\}$ and denoted by $obs(\mathbf{c})$ the noisy observation of $\mathbf{c}$. According to [3], the following steps are carried out iteratively during decoding:

$$\textit{Horizontal step:}\quad w_{m,n} = f(v_{m,1}, \ldots, v_{m,n-1}, v_{m,n+1}, \ldots) \tag{1a}$$

$$\textit{Vertical step:}\quad v_{m,n} = u_n \prod_{i \neq m} w_{i,n} \tag{1b}$$

In eqn. 1, $v_{m,n}$ is initialised to $u_n = \Pr\{c_n = +1|obs(\mathbf{c})\}/\Pr\{c_n = -1|obs(\mathbf{c})\}$. The horizontal step in eqn. 1*a* is regarded as the local decoder of codes formed by $\{v_{m,n}\}$. The vertical step (eqn. 1*b*) is regarded as the global parallel decoder for multiple concatenated codes. Considering the implementation issue of eqn. 1, the multiplication in eqn. 1*b* can be implemented by addition of the exponent indexes in the log-domain (see discussion below). The *f*-function in eqn. 1*a* is evaluated as

$$f(A) = A \qquad f(A,B) = \frac{f(A)\cdot B + 1}{f(A) + B}$$
$$f(A,B,C) = \frac{f(A,B)\cdot C + 1}{f(A,B) + C} \quad \cdots \tag{2}$$

which can be implemented using a lookup table. In the following, we introduce a more efficient implementation technique for evaluating eqn. 2.

Efficient implementation of f-function: During the decoding process, all variables are stored in log-domain in the form of

$$s^i \quad i \in I \quad I = 0, \pm 1, \pm 2, \ldots, \pm(2^{m-1} - 1)$$

with an *m*-bit quantisation scheme [3]. Assume $s > 1$, since $\{s^i, \forall\, i \in I\}$ and $\{(1/s)^i, \forall\, i \in I\}$ contain the same set of elements. We will store the indexes $\{i\}$. In the log-domain, eqn. 2 involves the following operations:

$$k = \log_s f(s^i, s^j) = \log_s \frac{s^i \cdot s^j + 1}{s^i + s^j} \tag{3}$$

which can be written as

$$k = \hat{k} + \delta \tag{4}$$

with

$$\hat{k} = \mathrm{sign}(i)\mathrm{sign}(j)\min(|i|, |j|) \tag{5}$$

and

$$\delta = \log_s\left(s^{-|i+j|} + 1\right) - \log_s\left(s^{-|i-j|} + 1\right) \tag{6}$$

(*Note*: In [4], the logarithm of *f*-function is called the 'box-plus' operation.) As suggested in [4], k can be approximated by $\hat{k}$, but this will incur a performance loss as shown in Fig. 1. This loss can be avoided as follows. With the above quantisation scheme, i and j are quantised to integers and so $\hat{k}$ is always an integer. Since k should also be quantised to an integer, we have

$$k = Q(\hat{k} + \delta) = \begin{cases} \hat{k} + 0 & \text{if } |\delta| \leq 0.5 \\ \hat{k} \pm 1 & \text{if } 0.5 < |\delta| \leq 1.5 \\ \hat{k} \pm 2 & \text{if } 1.5 < |\delta| \leq 2.5 \\ \quad\vdots & \qquad\vdots \end{cases} \tag{7}$$

where $Q(\cdot)$ is the quantisation function with threshold at the middle of two indexes. Thus k can be obtained by adding an integer correction term to $\hat{k}$. The range of this correction term is related to s as follows.

Starting with $s^{-|x|} + 1 < 2$, for $s > 1$ (see above), therefore,

$$0 < \log_s\left(s^{-|x|} + 1\right) < \log_s 2 \tag{8}$$

Combining eqn. 6 with eqn. 8,

$$-\log_s 2 < \delta < \log_s 2 \quad \text{or} \quad |\delta| < \log_s 2 \tag{9}$$

The range of δ is given as a function of s in eqn. 9. Table 1 summarises the consequence of eqns. 7 and 9.

In our simulation studies, we adopt $s = 3$ and $s = 2$ for $m = 4$ and $m = 5$ quantisation bit schemes, respectively. Based on eqns. 5 and 7 and Table 1, there are only two non-zero correction values ±1 for $s = 2$ and 3. Furthermore, by carefully listing all possible input-output values, eqn. 4 can be rewritten in the following forms.

Table 1: Possible values of quantised correction term with difference step size *s*

s	δ	Possible values of quantised correction term
$s \geq 4$	$\|\delta\| \leq 0.5$	0
$s \geq 1.5874$	$\|\delta\| \leq 1.5$	0, ±1
$s \geq 1.3915$	$\|\delta\| \leq 2.5$	0, ±1, ±2
⋮	⋮	⋮

For $m = 4$ and $s = 3$,

$$k = \log f(s^i, s^j) = \begin{cases} \text{sign}(\hat{k})(|\hat{k}| - 1) & |i| = |j| \text{ and } \hat{k} \neq 0 \\ \hat{k} & \text{otherwise} \end{cases} \tag{10}$$

For $m = 5$ and $s = 2$,

$$\begin{aligned} k &= \log f(s^i, s^j) \\ &= \begin{cases} \text{sign}(\hat{k})(|\hat{k}| - 1) & \big||i| - |j|\big| \leq 1 \text{ and } \hat{k} \neq 0 \\ \hat{k} & \text{otherwise} \end{cases} \end{aligned} \tag{11a}$$

except $k = \hat{k}$, when

$$|\hat{k}| = 1 \quad \text{and} \quad \max(|i|, |j|) = 2 \tag{11b}$$

We observed that the performance loss is negligible if eqn. 11*b* is ignored (i.e. use eqn. 11*a* only for all cases). Clearly, no multiplication or table searching is required in eqns. 10 and 11. The effect of using the correction term is shown in Fig. 1. The performance of the five-bit scheme with the correction term is very close to that of the ideal case.

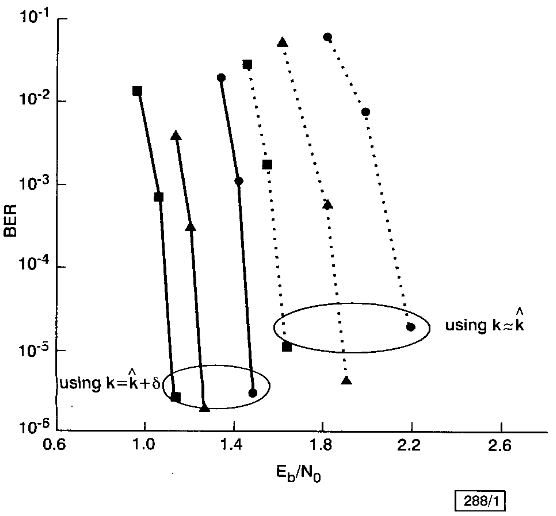

Fig. 1 *Performance of rate-1/2 LDPC code with information length = 30000 and iteration number = 18*

Serial (two-way) schedule is adopted in simulation (see [5 – 7])
■ no quantisation
▲ 5 bit quantisation
● 4 bit quantisation

Conclusion: We have introduced an efficient implementation technique for decoding the LDPC codes based on the likelihood ratio. The proposed method does not require multiplication or table searching, and thus leads to considerably reduced decoding cost.

Acknowledgment: This work was supported by the City University of Hong Kong under Grant 7100159.

6 March 2001
Electronics Letters Online No: 20010836
DOI: 10.1049/el:20010836

W.K. Leung, W.L. Lee, Angus Wu and Li Ping (*Department of Electronic Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong*)

Email: eeliping@cityu.edu.hk

References

1 GALLAGER, R.G.: 'Low density parity check codes', *IRE Trans. Inf. Theory*, 1962, **IT-8**, pp. 21–28
2 MACKAY, D.J.C., and NEAL, R.M.: 'Near Shannon limit performance of low density parity check codes', *Electron. Lett.*, 1997, **33**, (6), pp. 457–458
3 LI, PING, and LEUNG, W.K.: 'Decoding low density parity check codes with finite quantization bits', *IEEE Commun. Lett.*, 2000, **4**, (2), pp. 62–64
4 HAGENAUER, J., OFFER, E., MÉASSON, C., and MÖRZ, M.: 'Decoding and equalization with analog non-linear networks', *Eur. Trans. Telecommun.*, October 1999
5 LI, PING, and WU, K.Y.: 'Concatenated tree codes: a low complexity, high performance approach', *IEEE Trans. Inf. Theory*, 2000, **47**, (2)
6 KSCHISCHANG, F.R., and FREY, B.J.: 'Iterative decoding of compound codes by probability propagation in graphical models', *IEEE J. Sel. Areas Commun.*, 1998, **16**, (2), pp. 219–230
7 KSCHISCHANG, F.R., FREY, B.J., and LOELIGER, H.-A.: 'Factor graphs and the sum-product algorithm', submitted to *IEEE Trans. Inf. Theory* (available at http://www.comm.utoronto.ca/frank/factor/)

Optical orthogonal code design using genetic algorithms

C.K. Ho, S.W. Lee and Y.P. Singh

A construction technique for optical orthogonal codes (OOCs) using genetic algorithms (GAs) is proposed. The performance of the GA OOCs is compared to that of four existing OOCs. Results show that the GA OOCs have a lower probability of error.

Introduction: An optical orthogonal code (OOC) **C** is a collection of binary (0, 1) sequences that can be described by using four parameters, (n, w, λ_a, λ_c), where n is the length of the sequences, w is the Hamming weight of the sequences, λ_a is the auto-correlation constraint and λ_c is the cross-correlation constraint. Here, periodic correlation is considered. Let t_i represent the ith '1' in an OOC. A codeword of weight w is then represented in set notation as $\{t_1, t_2, \ldots, t_w\}$. OOCs are used as address code sequences in incoherent optical code division multiple access (CDMA) local area networks. An OOC has two important properties [1], i.e. the auto-correlation property as described by eqn. 1 and the cross-correlation property as described by eqn. 2 in the following:

$$\sum_{t=0}^{n-1} x_t x_{t+\tau} \leq \lambda_a \qquad 0 < \tau < n \tag{1}$$

$$\sum_{t=0}^{n-1} x_t y_{t+\tau} \leq \lambda_c \qquad \text{for any } \tau \tag{2}$$

where $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbf{C}$, $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathbf{C}$ and τ is an integer representing a time shift.

Code construction: The genetic algorithms have been designed to produce OOCs with any n, w, λ_a, and λ_c. The algorithms consist of the following steps:

Step 1: Encode each binary string codeword as a string of integers, where each integer corresponds to the positions of '1's in the codeword.

Step 2: Generate the initial population with weight = w and length = n. An individual is constructed by randomly generating w integers from a pool of integers containing 0 to $n - 1$. Let N represent the size of the population.

Step 3: Evaluate the fitness of each individual in the population using the following fitness function:

$$Fitness(x) = \begin{cases} 0 & \text{if } A_{x,x}(s) = 0 \\ \frac{1}{N} \sum\limits_{\substack{y=1 \\ y \neq x}}^{N} A_{x,y}(s) & \text{if } A_{x,x}(s) = 1 \end{cases} \tag{3}$$